Mississippi State University

Center for Air Sea Technology

DESIGN DOCUMENT for the MOODS DATA MANAGEMENT SYSTEM (MDMS) Version 1.0

Technical Note 06-94

August 1994

48pg

Prepared for: Naval Oceanographic Office, Stennis Space Center, Mississippi
Contract Number: NAS 13-330, Order No. 53

DESIGN DOCUMENT

and DATABASE SPECIFICATION

FOR THE

MASTER OCEANOGRAPHIC OBSERVATION DATA SET (MOODS) DATA MANAGEMENT SYSTEM (MDMS)

MDMS Version 1.0 (22 August 1994)

CONTRACT NO: NAS 13-330, Order No. 53

CONTRACT DELIVERABLE: 1b/1c

Prepared for:

NAVAL OCEANOGRAPHIC OFFICE
STENNIS SPACE CENTER, MS 39529

Prepared by:

Mississippi State University
Center for Air Sea Technology
Building 1103, Room 233
Stennis Space Center, MS 39529-6000

Acknowledgements

UNIX is a trademark of American Telephone and Telegraph (AT&T), Incorporated

SUN, SUNOS and SparcStation are trademarks of Sun Microsystems, Incorporated

Motif is a trademark of the Open Software Foundation

X-Windows is a trademark of the Massachusetts Institute of Technology

Empress is a trademark of Empress Software, Incorporated

UNIRAS ag/X Toolmaster is a trademark of Advanced Visual Systems, Incorporated

Contributors to the MDMS Software Design Document are:

Ms. Cheryl Cesario, Project Manager
Mr. Ramesh Krishnamagaru, Senior Software Engineer
Mr. Vishnu Mohan Das, Programmer
Mr. Micheal S. Foster, Graphics and Editing

Table of Contents

Page Number

LIST OF FIGURES:

MDMS SOFTWARE DESIGN DOCUMENT

1. SCOPE

1.1. Identification

Computer Software Configuration Item (CSCI): Master Oceanographic Observation Data Set (MOODS) Data Management System (MDMS)

Version: 1.0

Release Date: 22 August 1994

Contract No: NAS 13-330, Order No. 53

Contractor: Mississippi State University
Center for Air Sea Technology
J. H. Corbin, Director
Building 1103, Room 233
Stennis Space Center, MS 39529-6000
Telephone: (601) 688-2561
Facsimile: (601) 688-7100

Project Manager: Ms. Cheryl Cesario
Mississippi State University
Center for Air Sea Technology, Stennis Space Center, Mississippi 39529-6000. (MSU Proposal No. 93-3-467)
Telephone: (601) 688-7141
Facsimile: (601) 688-7100

1.2 Overview

The MOODS Data Management System (MDMS) provides access to the Master Oceanographic Observation Data Set (MOODS) which is maintained by the Naval Oceanographic Office (NAVOCEANO). The MDMS incorporates database technology in providing seamless access to parameter (temperature, salinity, soundspeed) vs. depth observational profile data. The MDMS is an interactive software application with a graphical user interface (GUI) that supports user control of MDMS functional capabilities.

1.3 Document Overview

The purpose of this document is to define and describe the structural framework and logical design of the software components/units which are integrated into the major computer software configuration item (CSCI) identified as MDMS, Version 1.0. The preliminary design is based on functional specifications and requirements identified in the governing Statement of Work prepared by the Naval Oceanographic Office (NAVOCEANO) and distributed as a request for proposal by the National Aeronautics and Space Administration (NASA). The content and format of this document are specified by Department of Defense (DOD)-STD 2167A (2.1).

Appendix A contains a glossary of terms used within this document. Appendix B is a listing of acronyms used within this document. The MDMS relational database specification is contained in Appendix C. Appendix D contains functional and design requirements for the MDMS relational database management system (RDBMS). Refer to Appendix E for a description of the MDMS development environment. Appendix F contains listings of relevant source code structures. Appendix G provides information about the MDMS configuration file.

2 REFERENCED DOCUMENTS

Note: The section(s) or subsection(s) of this document wherein a reference is cited appears as parenthetical information following each document listed in this section.

2.1. DOD-STD 2167A "Defense System Software Development", AMSC No. N4327, 29 Feb 88. (1.3)

2.2. Young, Douglas A., "The X Windows System Programming and Applications with Xt, OSF/Motif Edition", Prentice Hall, Englewood Cliffs, NJ, 1990. (3.1.1.1)

2.3. Jurkevics, Andrew, "Database Design Document for the Naval Environmental Operational Nowcast System, Version 3.5", Naval Oceanographic and Atmospheric Research Laboratory, Monterey, CA, 1 June 1992. (3.1)(3.1.1.4)

2.4. Documentation (Series) for the Empress Relational Database Management System, Version 6.X, Vol. A1-D1, Empress Software Incorporated, Greenbelt, MD, 1993. (3.1.1.4)

2.5. Users Manual and Reference Guides for UNIRAS ag/X Toolmaster Graphics Extensions Library, Version 6v3b, UNIRAS, Incorporated, Overland Park, KS, 1993. (3.1.1.1)

2.6. Coffin, Stephen, "UNIX System V Release 4: the Complete Reference", Osborne McGraw-Hill, New York, NY, 1990. (Useful reference for broad overview)

3 PRELIMINARY DESIGN

3.1 CSCI Overview

The Master Oceanographic Observation Data Set (MOODS) Data Management System (MDMS), a stand-alone system, is the major Computer Software Configuration Item (CSCI) to be developed by this project. Functional requirements, as identified by the sponsor, were not defined in the context of a modular approach to software development. Therefore, the tasks necessary to fulfill the functional requirements have been divided among/assigned to the internal Computer Software Components (CSC) for accomplishment. The top-level (simplistic) MDMS architecture is illustrated in Figure 1.

There are three principle CSC's within the MDMS structure:

a. **CSC-1: Graphical User Interface (GUI)** - incorporates initialization, window management, user interface and display functionality;

b. **CSC-2: Data Management Module (DMM)** - provides functional data management using relational RDBMS technology;

c. **CSC-3: Data Administration Module (DAM)** - incorporates administrative control of the application and is inaccessible to the general user.

The only access to the MDMS is via the **User-GUI** external interface which supports 1) application initialization, 2) user control of the MDMS using interactive techniques, 3) response/feedback to the user in the form of data display (graphical or numerical) and status indicators, 4) indirect access to data contained in the MOODS database, and 5) indirect access to files and data that are not resident within the MOODS database. The DMM accesses the MOODS database and data/configuration files through its external interfaces. The Naval Environmental Operational Nowcast System (NEONS)(2.3) controls access to the external MOODS database and has been modified to accommodate specific storage and retrieval requirements of the MDMS.

3.1.1 CSCI Architecture

Figure 1 illustrates the MDMS top-level (simplistic) module and its external interface architecture.

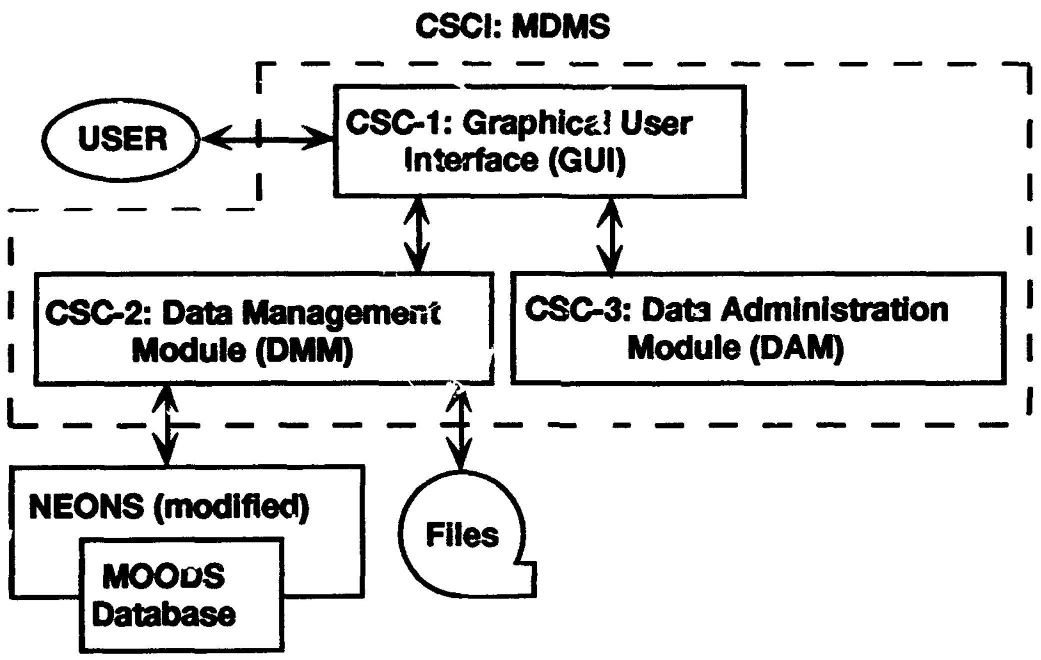

CSCI: MDMS
USER
CSC-1: Graphical User Interface (GUI)
CSC-2: Data Management Module (DMM)
CSC-3: Data Administration Module (DAM)
NEONS (modified)
MOODS Database
Files

Figure 1. Top-level modular structure of the MOODS Data Management System (MDMS). The MDMS is the Computer Software Configuration Item (CSCI). There are three Computer Software Components (CSC).

3.1.1.1 CSC-1: Graphical User Interface (GUI). The MDMS GUI supports and manages the link between the user and the MDMS. Through the GUI, the user exercises all available MDMS control options. The MDMS provides displays to the user for interpretation and interactive response. The GUI is integrated with the following non-developmental software:

a. **X-Windows** (proprietary): Manages all controls and display windows. The X-Windows client/server model supports remote user access to the MDMS using network protocol via the UNIX operating system. X-Windows operation is explained in detail in reference 2.2.

b. **Open Software Foundation (OSF) Motif Widget Toolkit** (proprietary): Provides a widely accepted standard inventory of window components (widgets) that is pleasing to the eye and easy to interpret. For more information on programming using the Motif Widget Toolkit, see reference 2.2.

c. **UNIRAS ag/X Toolmaster** (proprietary): Handles graphical representation and visualization of data displays. See reference 2.5 for detailed information concerning ag/X Toolmaster.

Functionally, the GUI:

a. Performs application initialization and event monitoring functions;
b. Exercises direct, centralized control over the DMM and DAM;
c. Monitors activities of the DAM and the DMM and its database/file interfaces;
d. Intercepts, interprets and routes user interactive commands;
e. Serves as the agent for communications between the DMM and the DAM;
f. Provides status information and feedback to the user;
g. Provides the windowing environment for visualization of data, display of control elements and intercepting user interactive commands;
h. Receives and interprets user input via the keyboard or mouse pointing device;
i. Incorporates visualization routines to plot data displays for viewing by the user.

3.1.1.2 CSC-2: Data Management Module (DMM). The MDMS DMM is primarily tasked with managing access to and communications with the external MOODS RDBMS via its NEONS (modified) interface. The DMM communicates with the GUI to pass information from the database and external files. It communicates with the DAM to enable MOODS Database Administrator (MDBA) control over application configuration. When a data request is received from the GUI, the DMM notifies NEONS to retrieve the data from the MOODS database. When the data has been received, the DMM informs the GUI and passes its location in memory. The DMM also receives data management instructions from both the GUI (user) and the DAM (MDBA). In turn, through NEONS,the DMM translates these instructions into Structured Query Language (SQL) commands to the RDBMS. The DMM is also responsible for allocating the internal memory space for data retrieved from the database or data destined for database ingestion.

3.1.1.3 CSC-3: Data Administration Module (DAM). The MDMS DAM is responsible for managing MDBA functions within the MDMS. MDBA functions are not accessible to the general user. These functions include importing data into the database and updating of MDMS application tables (classification codes, data source codes, instrument types) which are then made available for user interaction within the GUI.

3.1.1.4 NEONS (modified). The MOODS database is accessed via a modified version of NEONS, which performs high level management of the MOODS database. The specific modifications to NEONS for the MDMS allow use of additional keys to support optimal database query and data retrieval from the MOODS primary database tables (which contain the actual data). The MOODS database is a relational database. NEONS includes both the structural model for the database and the applications programmer interface (library routines) required to access the MOODS database tables using the American National Standards Institute (ANSI) standard SQL. NEONS was developed by the Naval Research Laboratory, Monterey, CA. A detailed description of NEONS is available in the NEONS design document (2.3). NEONS employs the proprietary Empress (2.4) relational data management system (RDBMS) for low-level management of the MOODS database.

3.1.1.5 USER-GUI (USER/CSC-1) External Interface. The USER-GUI interface provides user control of the MDMS. The interface is implemented by means of a pointing device (mouse) which is employed by the user to pass signal instructions (events) to the application, and the keyboard which is used to pass text information to the application.

3.1.1.6 GUI-DMM (CSC-1/CSC-2) Internal Interface. The GUI-DMM internal interface consists of all facilities for communication between the GUI and the DMM. The GUI-DMM interface is an abstraction employed to group those actions which pass information between the two modules. Actual communication between the GUI and the DMM is usually accomplished directly by the GUI and DMM functions involved.

3.1.1.7 GUI-DAM (CSC-1/CSC-3) Internal Interface. The GUI-DAM internal interface consists of all facilities for communication between the GUI and the DAM. The GUI-DAM interface embodies the same functionality and abstract character as the GUI-DMM interface.

3.1.1.8 DMM-DAM (CSC-2/CSC-3) Internal Interface. There are only two instances where the DMM and DAM communicate directly with each other. These instances are discussed in Section 4.2.2. Otherwise, the DMM and DAM communicate indirectly via the GUI.

3.1.1.9 DMM-NEONS External Interface. The DMM-NEONS external interface consists of all NEONS library functions that are integrated within the MDMS. These functions generate SQL commands to the Empress RDBMS, returning status indicators and performing two-way data transferal, when successfully executed.

3.1.1.10 DMM-FILES External Interface. The DMM-FILES external interface consists of all functions within the DMM which result in direct reading from or writing to files within the operating system's file management facility.

3.1.2 System States and Modes

The MDMS is an interactive software application. The application is always in an event-driven state. As with all event-driven software applications, the MDMS may assume either of two execution states (modes): (1) processing mode and (2) rest (idle) mode. The MDMS default state is the rest mode. When not processing data in response to user input, the application automatically reverts to the rest mode (event monitoring loop) and awaits the next user command or input.

3.1.3 Memory and Processing Time Allocation

The interactive, event-driven nature of the MDMS precludes a quantitative description of memory allocation and processing time among MDMS CSC's. It is therefore considered sufficient to state that the MDMS responds to user-generated commands by allocating memory, swap space and processor time within the limitations imposed by available memory, process loading and UNIX operating system constraints.

3.2 CSCI Design Description

The CSCI (MDMS) consists of three CSC modules with functionality as described above. Functional requirements for the CSCI, as stated by the sponsor, could not always be accomplished wholly and completely within a single CSC. For instance, the final display of profile data requires the cooperative contribution of both the GUI and the DMM modules. The remainder of this section identifies MDMS requirements, by CSC, and describes the software design employed to achieve the required functionality. The CSCI incorporates all MDMS functional requirements (MFR) and design requirements (MDR) as presented in Section 7 of this document. In addition, the CSCI achieves specific design requirements which are not accomplished by any CSC (MDR1 through MDR4).

3.2.1 Graphical User Interface (GUI) (CSC-1)

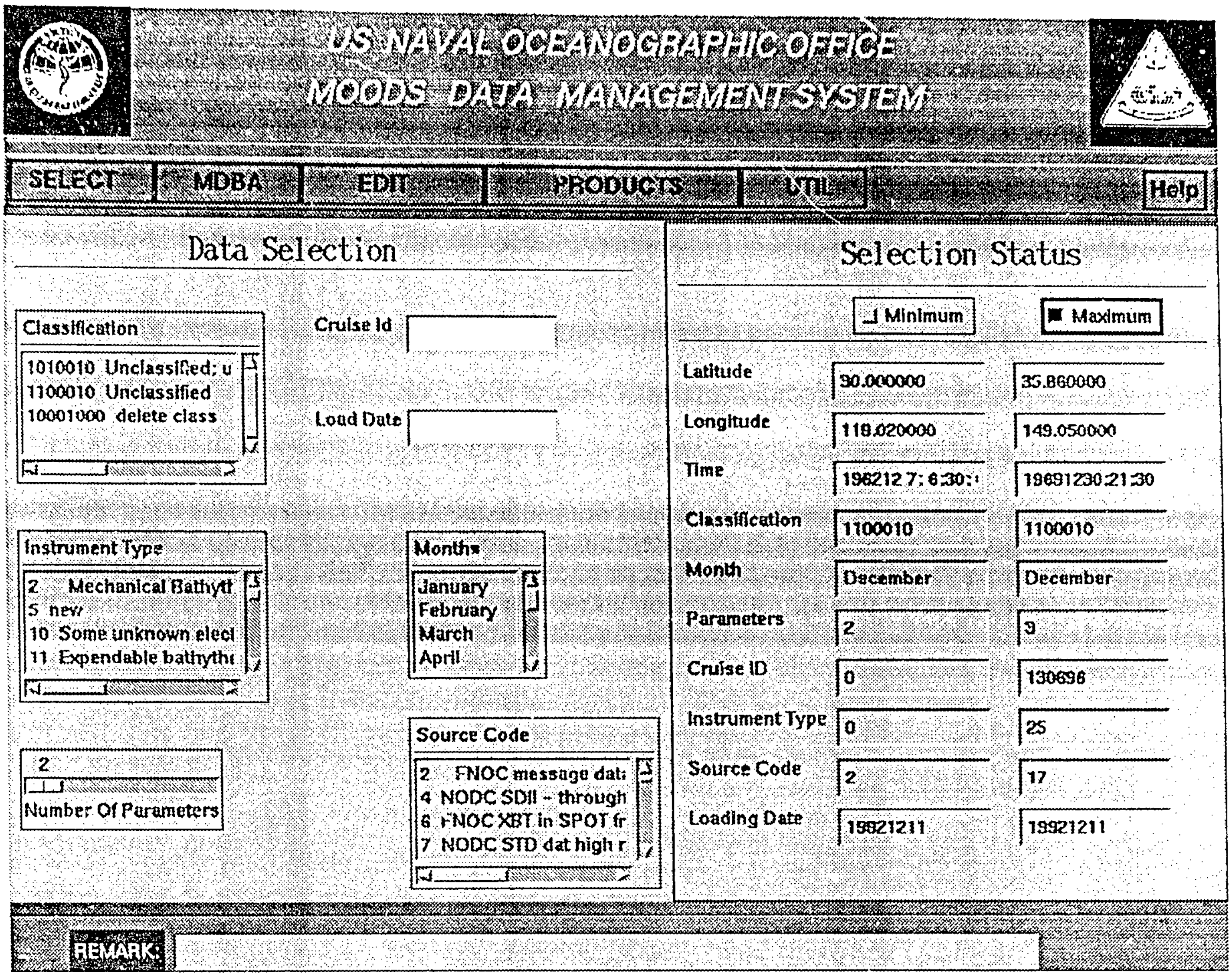
CSC-1 is responsible for providing the visual display link between the user and the MDMS main interactive display. The design for the MDMS main interactive display window is illustrated in Figure 2. The GUI is the ultimate destination of all visible computational results achieved by the CSCI and its subordinate CSC's. The DMM (CSC-2) performs data management chores for the GUI and DAM. The DAM (CSC-3) performs MDBA applications control and maintenance functions. The GUI and its subordinate components achieve the following specific functional and design requirements, either wholly or in concert with other CSC's (see Section 7):

MDMS Functional Requirements (MFR)

MFR1, MFR2, MFR3, MFR4, MFR5, MFR6, MFR7, MFR8, MFR9, MFR10 MFR11, MFR12, MFR13, MFR14, MFR15, MFR17, MFR18, MFR19, MFR 20, MFR21, MFR22, MFR23, MFR24, MFR26, MFR30, MFR32, MFR33.

MDMS Design Requirements (MDR)

MDR8



Data Selection

Classification
1010010 Unclassified; u
1100010 Unclassified
10001000 delete class

Cruise Id

Load Date

Instrument Type
2 Mechanical Bathyt
5 new
10 Some unknown elect
11 Expendable bathythe

Months
January
February
March
April

2
Number Of Parameters

Source Code
2 FNOC message dat
4 NODC SDII - through
6 FNOC XBT in SPOT fr
7 NODC STD dat high r

Selection Status

	Minimum	■ Maximum
Latitude	30.000000	35.860000
Longitude	118.020000	149.050000
Time	1962127; 6:30:	19691230:21:30
Classification	1100010	1100010
Month	December	December
Parameters	2	9
Cruise ID	0	130698
Instrument Type	0	25
Source Code	2	17
Loading Date	19921211	19921211

Figure 2. Illustration of the MDMS Graphical User Interface (GUI) (CSC-1) display screen. The GUI is the origin and final destination for all interactive coordination between the user and user-controllable processes resident in the Data Management Module (DMM) (CSC-2), the Data Administration Module (DAM) (CSC-3) and their internal/external interfaces.

3.2.2 Data Management Module (DMM) (CSC-2)

The Data Management Module (DMM) (CSC-2) controls the acquisition and distribution of data within the MDMS CSCI. The user has no direct interaction with the DMM. The DMM and its sub-level components achieve the following specific functional and design requirements, either wholly or in concert with other CSC's (see Section 7):

MDMS Functional Requirements (MFR)

MFR13, MFR14, MFR15, MFR17, MFR18, MFR19, MFR20, MFR21, MFR22, MFR23, MFR24, MFR27, MFR28, MFR29, MFR30, and MFR31.

MDMS Design Requirements (MDR)

MDR5, MDR6, and MDR7.

3.2.3 Data Administration Module (DAM) (CSC-3)

The DAM provides tools that enable application maintenance by the MDBA. There are no specific design requirements for the DAM. The DAM and its sub-level components achieve the following specific functional requirements, either wholly or in concert with other CSC's (see Section 7):

MDMS Functional Requirements (MFR)

MFR12, MFR13, and MFR15.

4 DETAILED DESIGN

Figure 3 illustrates the functional connectivity within the MDMS and forms the basis for the detailed design of the MDMS CSCI. The preliminary design (see Section 3) provides a broad overview of the framework and functional design of the MDMS CSCI and its three major CSC's [Graphical User Interface (GUI), Data Management Module (DMM) and Data Administration Module (DIM)]. In this section, each CSC and its computer software units (CSU) will be described in greater detail.

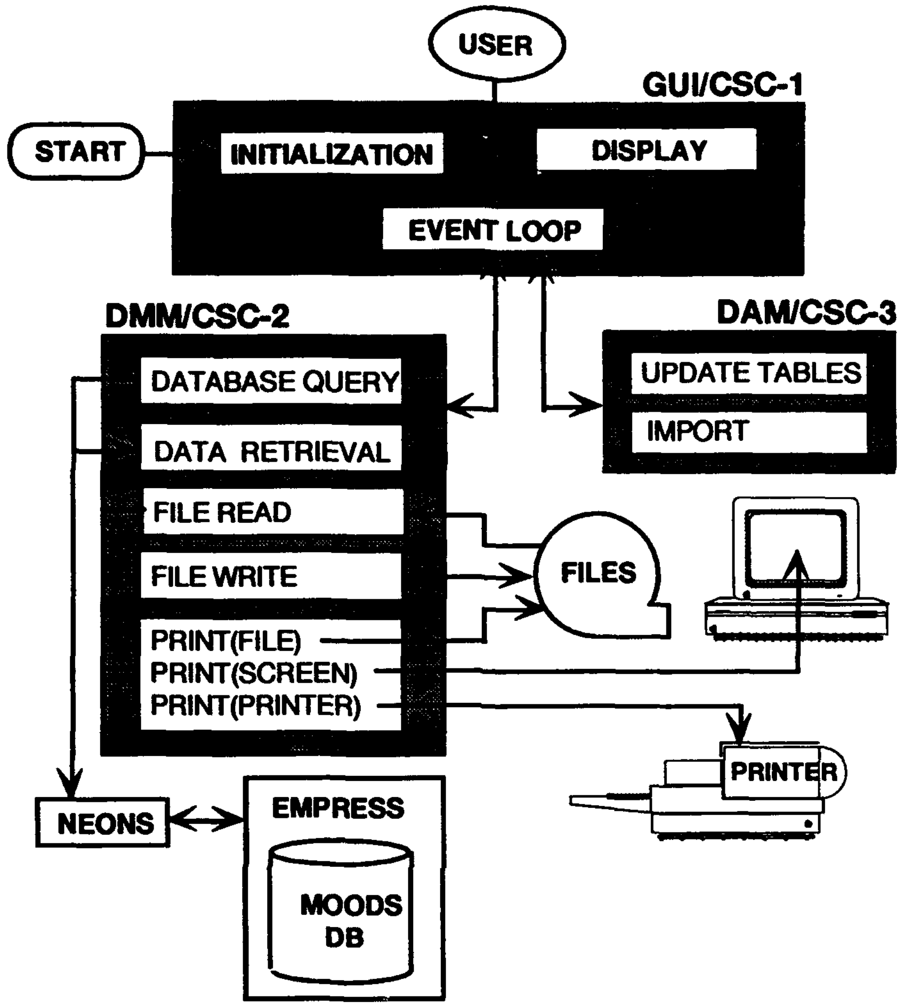


Figure 3. MDMS detail design and connectivity.

4.1 CSC-1 - The MDMS Graphical User Interface (GUI)

The MDMS GUI (CSC-1) embodies all facilities for user interaction. Functions performed by the GUI include:

a. Application initialization;
b. Event monitoring;
c. Window management;
d. Communicating with the user and the Data Management and Data Administration modules;
e. Creation and display of data plots from data received from the DAM;
f. Processing and interpretation of user commands via the pointing device (mouse) and the keyboard;
g. Providing visual feedback in response to user input and internal messages;
g. Management of the user "Help" facility;
g. Application shutdown.

4.1.1 Initialization

The Initialization CSU receives control when the operating system releases control to the application. Functions performed by the Initialization CSU include:

a. Parsing and interpreting any command line arguments that identify the default configuration filename by the function **parseArg()**;
b. Establishing a connection to the X server, initializing the Intrinsics layer and obtaining a TopLevelShell widget by the function **XtInitialize()**;
c. Obtaining pointers to the Xlib Display and Screen structures using the functions **XtDisplay()** and **XtScreen()**;
d. Reading the MOODS configuration file;
e. Opening the database using the **db_start()** function;
f. Creation and display of the MDMS Main Window, its window tree and data structures including default range values obtained from the configuration file via the function **CreateTree()**;
g. Initializing internal MOODS data structures via function **InitMoodsStruct()**;
h. Establishing user access privileges.
i. Executing the **XtMainLoop()** event monitoring function which dispatches all subsequent events to registered event handlers and/or callback functions.

4.1.1.1 MDMS Initialization Specification/Constraints

The design criteria for the MDMS Initialization CSU are listed in Section 7, "Requirements Traceability". This CSU is constrained by the following software-specific requirements:

a. X-Windows and OSF Motif must be installed and accessible;
b. The Empress RDBMS must be installed and accessible;
c. UNIRAS ag/X Toolmaster must be installed and accessible;
d. NEONS must be installed and accessible;
e. A MOODS database must be available.

4.1.1.2 MDMS Initialization Design

The MDMS Initialization design adopts the standard X-Windows/OSF Motif protocol. The **XtMainLoop()** function ultimately assumes control after all initialization procedures have been accomplished, including display of the Main Window. **XtMainLoop()** implements an infinite event-monitoring loop which executes (dispatches messages to) registered event handling and/or callback functions in response to the occurrence of events.

4.1.2 MDMS Main Window

The Main Window CSU is illustrated in Figure 2. It is subdivided into five areas:

a. The title area;
b. The menubar which allows access to user-selectable menu items via pulldown menus;
c. The "Data Selection" area which provides access to most configuration parameters required to identify MOODS datasets and/or subsets;

d. The "Selection Status" area which displays range limits (minimum and maximum) for parameters defined within the "Data Selection" area and for the "Select: Region" and "Select: Time" menu items;
e. The "Remark" textbox which communicates important messages, such as errors, to the user.

The Main Window is created and displayed during application initialization. All MDMS features are initiated via the Main Window or its subordinate windows (pop-ups, textboxes, listboxes, etc.) which appear in response to user interaction with Main Window selection features.

4.1.2.1 MDMS Main Window Specification/Constraints

The design criteria for the MDMS Main Window CSU are listed in Section 7, "Requirements Traceability". This CSU is constrained by the following requirements:

a. The X-server must be opened and available for display;
b. UNIRAS ag/X Toolmaster must be active for handling graphics functionality;
c. The Empress RDBMS must be active and the MOODS database must be opened and available for reading.

4.1.2.2 MDMS Main Window Design

The MDMS Main Window design employs the X, Motif, NEONS, and UNIRAS ag/X Toolmaster libraries. The design is implemented during initialization by the function **CreateTree()**.

4.1.3 GUI Functionality

After the MDMS Main Window has been displayed, the user is free to interact with it using the mouse and keyboard. The GUI translates all mouse and keyboard events into calls to event handlers and/or callback functions. If the event requires retrieval/ingestion of data from/to the MOODS database, the GUI calls upon a function within the DMM. If the event requires MDBA intervention or a security check, the GUI calls upon a function within the DAM.

4.1.3.1 The Menubar

The menubar consists of five pull down menu headers (**Select, MDBA, Products, Util**), each of which offers two or more subordinate menu items, plus the **Help** button which provides access to the on-line help facility. When clicked, a menu header is programmed to display its subordinate menu items, which themselves may be selected by clicking. The reaction of the application to selection of a menu item depends upon the functionality of the particular menu item. Selections made via the menubar provide access to all options that are not directly associated with establishing the criteria for data retrieval.

4.1.3.1.1 Select: Region

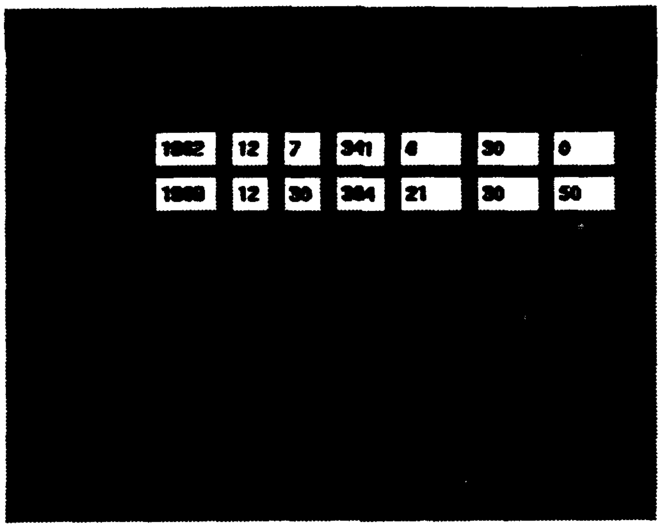
When the "Region" menu item is clicked, the function **regionButtonPress()** is executed and the window illustrated in Figure 4 is displayed. This window offers three options for selecting the boundaries of a geographical region. The "Region List" contains a listing of user-selectable pre-defined regions. When a pre-defined region is selected from the "Region List", the region boundary is displayed within the map window as a dashed rectangle, and, the labeled textboxes within the "Region Coordinates" widget are modified to hold the latitude and longitude coordinate limits of the region. When a region is selected by dragging the mouse cursor across an area of the map display, the "Region Coordinates" textboxes are modified to hold the latitude and longitude coordinates of the region. Finally, a region may be defined by selecting individual textboxes within the "Region Coordinates" widget and entering replacement values from the keyboard. If greater resolution is required, the "Zoom" button located below the map will produce a pop-up window containing an enlarged image of the region defined on the map. The "File" pulldown menu provides menu items to exit the display and to reset the coordinates to their original values. Selecting the "Exit" menu item destroys the window and sends the latitude and longitude boundary coordinates to the "Selection Status" board within the MDMS Main Windc w.

35.000000
-45.000000
5.000000
-100.000000

Figure 4. The MDMS Region Selection Window. This window appears in response to selecting the "Region" menu item via the "Select" pulldown menu.

4.1.3.1.2 Select: Time

When the "Time" menu item is clicked, the function **timeButtonPress()** is executed and the "Time Selection" window illustrated in Figure 5 is displayed. This window offers options for selecting the time constraints for a MOODS database query. If the "Start" button is activated, the adjacent row of text widgets is also activated for input of minimum date/time values. If the "End" button is activated, the adjacent text widgets are activated for entry of maximum date/time values. Date entries may also be changed using the up or down arrow switch buttons to the right of the vertically arrayed "YEAR", "MONTH", "DAY", and "JULIAN" textboxes. These arrow switches are manipulated by clicking. Changes made using the arrow switches are applied to the activated "Start" or "End" row of textboxes. Clicking the "Exit" button closes the "Time Selection" window and transfers the values to the maximum and minimum Date/Time textboxes within the "Selection Status" board of the MDMS Main Window.



1992 12 7 341 6 30 0

1993 12 30 364 21 30 50

Figure 5. The MDMS Time Selection Window. This window appears in response to selecting the "Time" menu item via the "Select" pulldown menu.

4.1.3.1.3 Select: Exit

When the "Exit" menu item is clicked, the function **exitProgram()** is executed. Clicking the "Exit" menu-item closes the MDMS main display screen and returns the user to the system command line prompt.

4.1.3.1.4 MDBA:Import (MDBA Only)

During initialization, the function **createDbaTools()** determines if the user is the MDBA. The source code...

```
if (!(isDba)) { XtSetArg (wargs[n], XmNsensitive, False);
```

sets the "Import" menu item's sensitivity argument to "False", thereby disallowing data import by users other than the MDBA. When the MDBA selects the "Import" menu item from the "MDBA" pull-down menu, the function **createImportExportDialog()** is executed with the **importexport->type** variable set to **IMPORT** and the "Input File" pop-up window (Figure 6) is displayed. The "Input File" pop-up window contains a textbox for entry of the file name (including path, if needed) and "Ok" and "Exit" buttons. To be ingested, the file contents must be in the "MOODS Admin" format. Clicking the "Ok" button executes the function **importexportOkSelect()** which, in turn, calls the DMM function **readIngest_file()** to read the file and ingest the data into the database. Clicking the "Exit" button executes the function **importexportExitSelect()** which destroys the "Import File" pop-up window and returns to the **XtMainLoop()** function without performing any data ingestion operations.

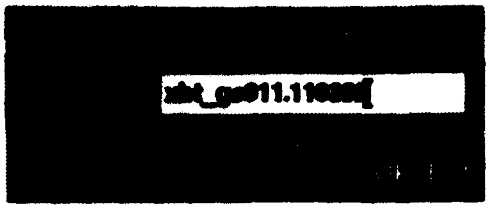


Figure 6. MDBA Filename Entry Dialog Window for Importing Files.

4.1.3.1.5 MDBA:Export

The MOODS export facility is available to all users possessing a level of access appropriate for the data to be exported. When the user selects the "Export" menu item from the "MDBA" pull-down menu, two additional menu items appear, "Export Admin" and "Export IDEAS". If "Export Admin" is selected, the function **createImportExportDialog()** is executed with the **importexport->type** variable set to **EXPORT** and the "Export File" pop-up window (Figure 7.a) is displayed. If "Export IDEAS" is selected, **createImportExportDialog()** is executed with the **importexport->type** variable set to **EXPORT_IDEAS** and the "Export IDEAS File" pop-up window (Figure 7.b) is displayed. Both export pop-up windows contain a textbox for entry of the export file name (including path, if needed), and "Ok" and "Exit" buttons. Clicking the "Ok" button executes the function **importexportOkSelect()** which, in turn, calls the DMM function **exportData2File()** which writes the data to the named file in either MOODS "Admin" or MOODS "IDEAS" format. The format depends upon the value of **importexport->type** (**EXPORT** or **EXPORT_IDEAS**). Clicking the "Exit" button executes the function **importexportExitSelect()** which destroys the file export pop-up window and returns to the **XtMainLoop()** function without performing any data export functions.

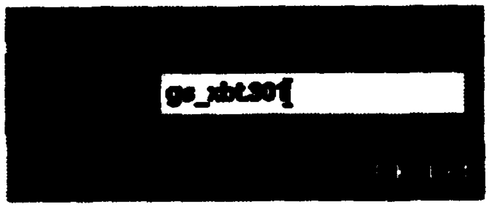


Figure 7.a. MDBA Filename Entry Dialog Window for MOODS "Admin" Export.

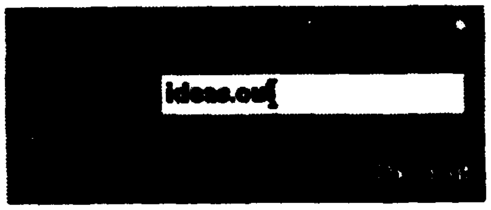


Figure 7.b. MDBA Filename Entry Dialog Windows for MOODS "IDEAS" Format.

4.1.3.1.6 MDBA:Update Table (MDBA Only)

The "Update Table" menu item is a tool for use of the MDBA only. During initialization, the function **createDbaTools()** determines if the user is the MDBA. The source code...

```
if (!(isDba)) { XtSetArg (wargs[n], XmNsensitive, False);}
```

sets the "Update Table" menu item's sensitivity argument to "False", thereby disallowing data table updates by users other than the MDBA. When selected, the "Update Table" menu item produces a submenu with three selectable menu items: "Update Instrument"; Update Source"; and "Update Classification". The GUI functions respectively responsible foi these options are **updateClassification()**, **updateInstrument()**, and **sourceUpdate()**, producing the pop-up windows shown in Figures 8.a, 8.b, and 8.c. Each of these functions executes the DAM function **changeTable()** to "ADD", "DELETE", or "UPDATE" the contents of the specified qualitative code. **changeTable()**, in turn, calls the DMM function **moodsUpdateQltvInfo()** which updates the MOODS database qualitative information table and makes the appropriate entry in the MOODS Log.

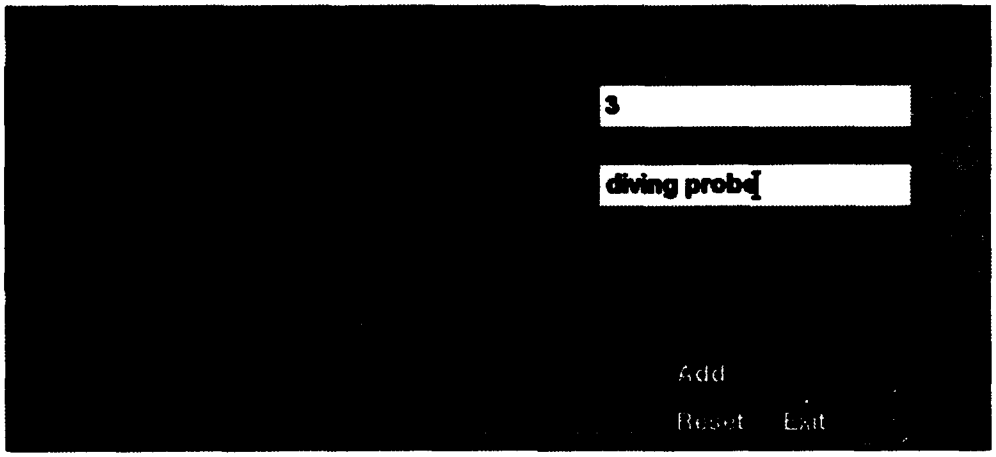


Figure 8.a. MDBA Update Instrument Pop-up Window.

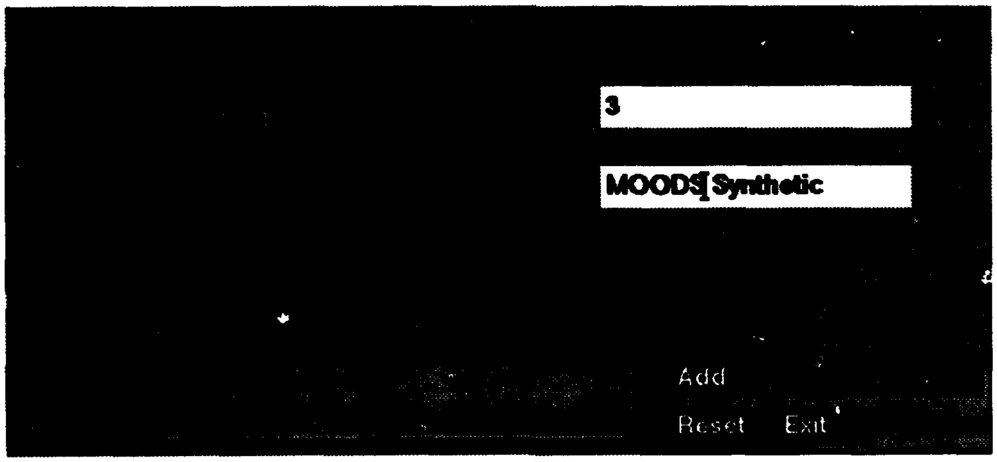
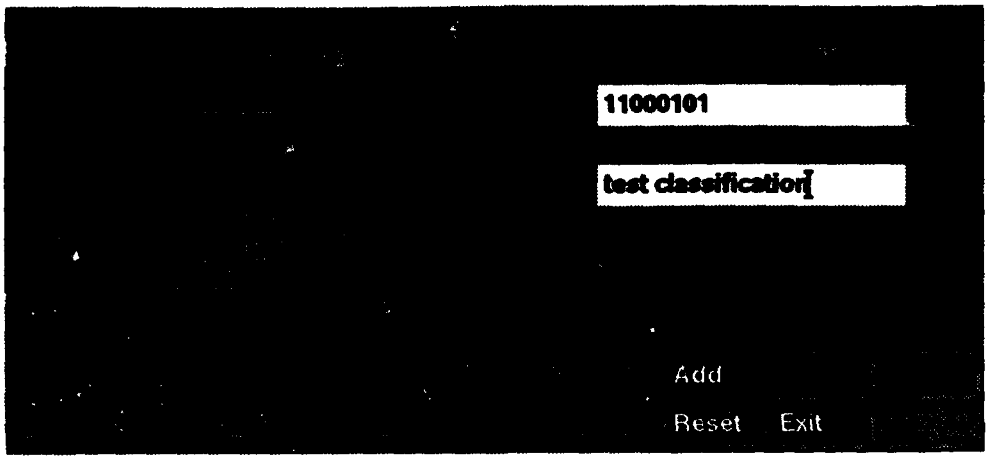


Figure 8.b. MDBA Update Source Pop-up Window.


11000101

test classification

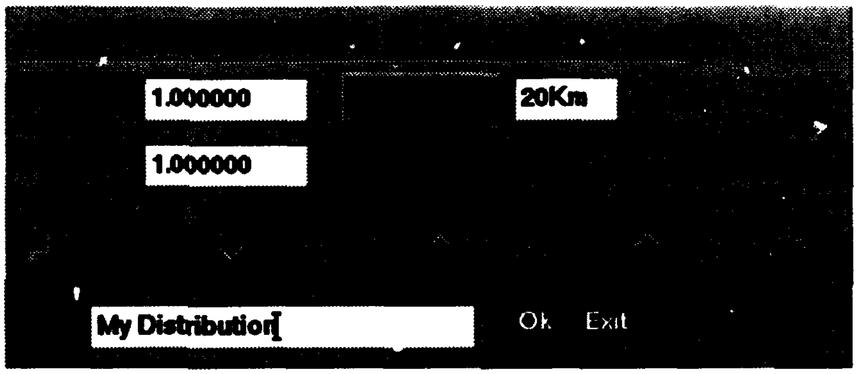
Figure 8.c. MDBA Update Classification Pop-up Window.

4.1.3.1.7 PRODUCTS: Distribution

When the user clicks the "Distribution" menu item from the "Products" pulldown menu, the GUI function **selectDistribution ()** is executed. **selectDistribution()** produces the MOODS "Distribution Plot" pop-up window shown in Figure 9, which allows user definition of data distribution plotting options ("Distribution", "Histogram" or "Accumulation"). This window also offers choices for latitude/longitude and coastline resolution, display to screen or printer, and (optionally) the inclusion of a title. Relevant GUI callback functions and their roles in generating MDMS Products:Distribution displays are as follows:

plotTypeTogglePress() - upon selection (click) of the "Distribution" toggle button.
histTypeTogglePress() -upon selection (click) of the "Histogram" toggle button.
acclTypeTogglePress() - upon selection (click) of the "Accumulation" toggle button.
devTogglePress() - upon user selection of display choice ("Screen" or "Printer").
choose3Km(), **choose8Km()**, or **choose20KM()** - upon selection of the type of coastline to be drawn in conjunction with the plot.
exitDistribution() - upon selection (click) of the "Exit" pushbutton.
enterLatDeg() - supports text entry of latitude (east/west) square size.
enterLonDeg() - supports text entry of longitude (north/south) square size.
enterTitle() - supports text entry of an optional plot title.
executePlotDistribution() - upon selection (click) of the "Ok" button and calls the appropriate routine based on the plot type selected.
exposeDistribution() - exposes the "Distribution" plot drawing area window and its contents.
destroyDialog() - destroys the pop-up window upon exiting.
exposeHistogram() - exposes the "Histogram" plot drawing area window and its contents.
exposeAccl() - exposes the "Accumulation" plot drawing area and its contents.

These functions will, in turn, call a variety of associated functions within the DMM to obtain the appropriate data from the database. The data query is constructed from information contained in the "Selection Status" board (Section 4.1.3.3). Figures 10.a and 10.b are example distribution/accumulation and histogram plots.



1.000000

20Km

1.000000

My Distribution

Figure 9. Distribution Plot Definition Window.

Figure 10.a. Profile Distribution/Accumulation Plot Window.

Figure 10.b. Histogram Plot of Profile Distribution.

4.1.3.1.8 PRODUCTS: Depth Vs. Parm

When the "Depth Vs Parm" menu item is clicked, the function **selectParmvsDepth()** is executed. **selectParmvsDepth()** produces an adjacent pulldown submenu for choosing Depth vs. Temperature, Depth vs. Salinity; or Depth vs. Sound Speed. Selection of either submenu item produces the appropriate depth vs. parm definition window for that option. Figure 11 illustrates the depth vs. temperature parm definition window (The depth vs. salinity and depth vs. sound speed definition windows are labeled "Salinity" and "Sound Speed", respectively; otherwise, they are identical in appearance and functionality to the depth vs. temperature window). Relevant GUI functions and their roles in generating MDMS Products:Depth vs. Parm displays are as follows:

sndSpdSelect() - when Sound Speed parameter has been chosen as the composite plot.
tempSelect() - when Temperature parameter has been chosen as the composite plot..
salinitySelect() - when Salinity parameter has been chosen as the composite plot.
tempvsSalinitySelect() - when Temp vs Salinity has been chosen as the composite plot.
countSizeEdit() - supports text entry of maximum number of MOODS profiles.
minXEdit() - supports text entry of minimum value for X-axis.
maxXEdit() - supports text entry of maximum value for X-axis.
minYEdit() - supports text entry of minimum value for Y-axis.
maxYEdit() - supports text entry of maximum value for Y-axis.

titleEdit() - supports text entry of a graph title.
exitParmVsDepth() - when the "Exit" pushbutton has been selected.
compDevTogglePress() - when the device selection (printer/screen) toggle button has been clicked.
exposeComposite() - to expose the composite drawing area.

When the "Ok" button is pressed, the function **executeCompositePlot()** calls the DMM function **commonMoodsRead()** which retrieves appropriate data from the MOODS database. **executeCompositePlot()** then calls the GUI function **drawComposite()**, which draws and exposes the composite parameter vs. depth window and its contents. Figure 12 illustrates the "Temperature vs. Depth" composite plot. Other composite plots are similar.

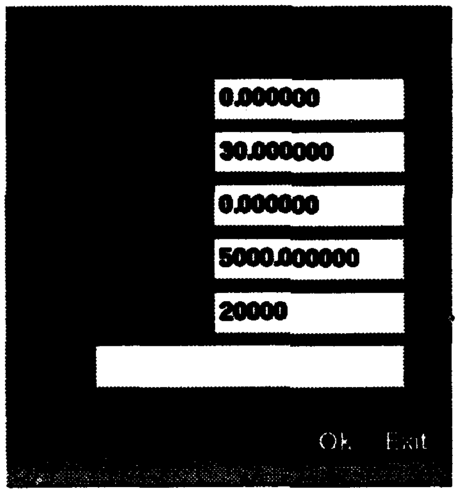

0.000000
30.000000
0.000000
5000.000000
20000

Figure 11. Depth vs. Temperature Definition Window.

Figure 12. Depth vs. Temperature Plot Example.

4.1.3.1.9 PRODUCTS: Print Header

The function **selectPrintHeader()** creates the dialog pop-up window, Figure 13, in response to selection of the "Print Header" menu item from the "Products" pulldown menu. **selectPrintHeader()** calls the function **outputFormatDialog()** to perform the actual creation of the dialog pop-up window. **outputFormatDialog()** registers callback functions **fileTogglePress()**, **screenTogglePress()** and **printTogglePress()** which process user selection of an output device. **fileTogglePress()** calls the function **filenameDialog()** to create a secondary pop-up dialog window for retrieval of a user-entered filename by the callback function **readFilename()**. Selection of the "Exit" button is processed by the callback function **exitPrintHeader()**. The callback function **executePrintHeader()** is registered (indirectly, via the **outputFormatDialog()** argument **proc()**) to process selection of the "OK" button. **executePrintHeader()** calls the DMM function **commonMoodsRead()** to retrieve the appropriate data from the database and the DMM function **moodsLogMesgEntry()** to make the required log entry. The screen, file or printer output from the "Print Header" menu item is in MOODS ADMIN format. Screen output is displayed by making a system call invoking the UNIX vi text editor.

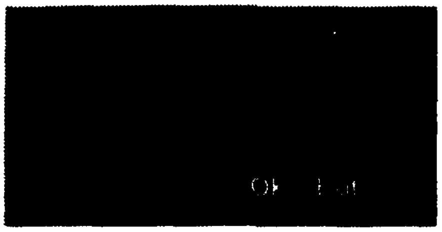


Figure 13. Print Header Output Selection Window.

4.1.3.1.10 PRODUCTS: Print Log (MDBA Only)

The "Print Log" menu item responds only to the MDBA. The "Print Log" dialog window, Figure 14, is produced by the callback function **selectPrintLog()** which calls the function **printLogLayout()** to construct the necessary window components and register the necessary callback functions as follows:

> **readFunction()** - to capture the MDMS transaction function type(s) selected from the scrollable "Functions" list.
> **readUserName()** - to capture the name of the user of interest to the MDBA.
> **readMinDate()** - to capture the minimum date as entered in the "Min Date" field.
> **readMaxDate()** - to capture the maximum date as entered in the "Max Date" field.
> **to_fileTogglePress()** - to process MDBA selection of the "File" output option.
> **executeLogData()** - to process MDBA selection of the "OK" button.
> **resetLogStruct()** - to process MDBA selection of the "Reset" button.
> **quitPrintLog()** - to process MDBA selection of the "Exit" button.

fileTogglePress() calls the function **filenameDialog()** to create a secondary pop-up dialog window for retrieval of a user-entered filename by the callback function **readFilename()**. **executeLogData()** calls the DMM function **moodsLogData()** to retrieve log entries that meet the criteria established by MDBA entries in the "Print Log" dialog window and output the data to screen, printer or file (see Figure 15).

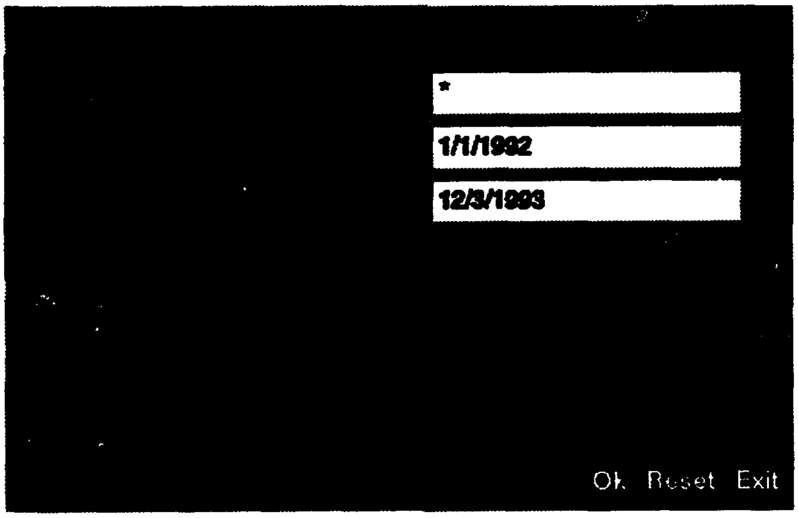
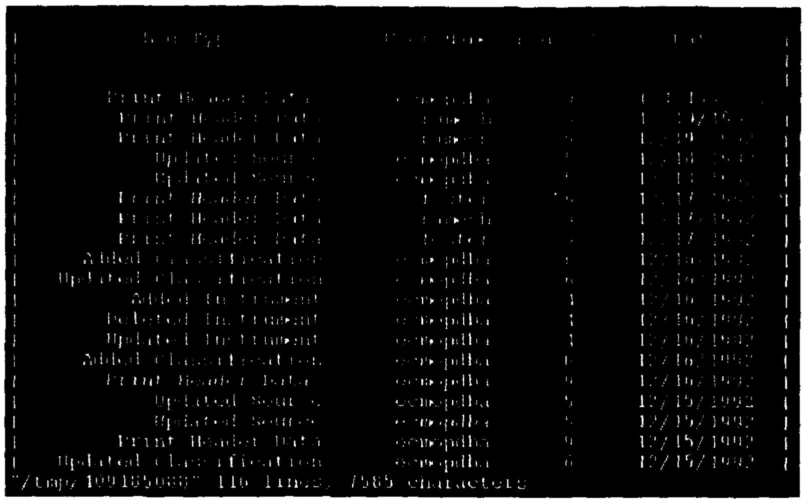


Figure 14. Print Log Definition Window.



```
|        [illegible]                [illegible]                   [illegible]      |
|                                                                                  |
|                                                                                  |
|      Print Header Data          [illegible]          [illegible]  [illegible]     |
|       Print Header Data          [illegible]          [illegible]  [illegible]    |
|       Print Header Data          [illegible]          [illegible]  [illegible]    |
|          Updated Source        cemopella          [illegible]  [illegible]        |
|          Updated Source        cemopella          [illegible]  [illegible]        |
|       Print Header Data          [illegible]          [illegible]  [illegible]    |
|       Print Header Data          [illegible]          [illegible]  [illegible]    |
|       Print Header Data          [illegible]          [illegible]  [illegible]    |
|    Added Classification        cemopella          [illegible]  12/16/1992         |
|  Updated Classification        cemopella          6     12/16/1992               |
|         Added Instrument        cemopella          1     12/16/1992               |
|       Deleted Instrument        cemopella          1     12/16/1992               |
|       Updated Instrument        cemopella          1     12/16/1992               |
|    Added Classification        cemopella          6     12/16/1992               |
|      Print Header Data          cemopella          9     12/16/1992               |
|          Updated Source         cemopella          5     12/15/1992               |
|          Updated Source         cemopella          5     12/15/1992               |
|       Print Header Data         cemopella          9     12/15/1992               |
|  Updated Classification        cemopella          6     12/15/1992               |
"/tmp/1091850880" 116 lines, 7585 characters
```

Figure 15. Print Log Screen Output Example.

4.1.3.1.11 PRODUCTS: Page Summary

The "Page Summary" pop-up dialog menu item, Figure 16, is created by the callback function **selectPageSummary()**. **selectPageSummary()** presents three page summary output options (screen, printer or file). Actual creation of the dialog window is accomplished by the function **outputFormatDialog()** which registers callback functions **fileTogglePress()**, **screenTogglePress()** and **printTogglePress()** to process user selection of an output device. **fileTogglePress()** calls the function **filenameDialog()** to create a secondary pop-up dialog window for retrieval of a user-entered filename by the callback function **readFilename()**. The function **executePageSummary()** is passed as an argument to **outputFormatDialog()** by **selectPageSummary()**. **executePageSummary()** processes retrieval of the appropriate data from the database via DMM function **commonHeaderRead()** and outputs the page summary to file, screen or printer. **outputFormatDialog()** reuses the callback function **exitPrintHeader()** to process selection of the "Exit" button. Figure 17 is an example of "Page Summary" screen output.

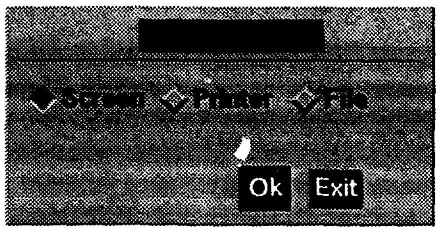
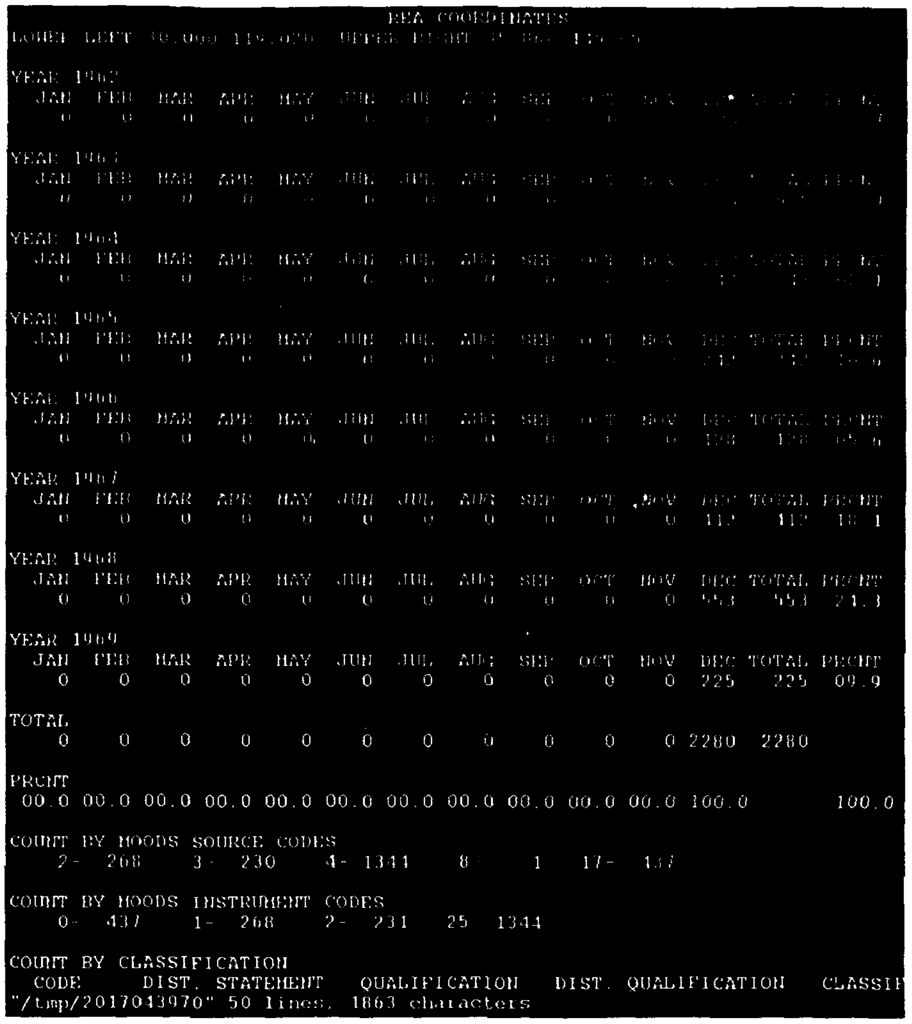


Figure 16. Page Summary Definition Window.



```
                                        REA COORDINATES
LOWER LEFT 30.000 119.070  UPPER RIGHT [illegible]
 
YEAR 1962
  JAN   FEB   MAR   APR   MAY   JUN   JUL   AUG   SEP   OCT   NOV   DEC TOTAL PRCNT
    0     0     0     0     0     0     0     0     0     0     0 [illegible]
 
YEAR 1963
  JAN   FEB   MAR   APR   MAY   JUN   JUL   AUG   SEP   OCT   NOV   DEC TOTAL PRCNT
    0     0     0     0     0     0     0     0     0     0     0 [illegible]
 
YEAR 1964
  JAN   FEB   MAR   APR   MAY   JUN   JUL   AUG   SEP   OCT   NOV   DEC TOTAL PRCNT
    0     0     0     0     0     0     0     0     0     0     0 [illegible]
 
YEAR 1965
  JAN   FEB   MAR   APR   MAY   JUN   JUL   AUG   SEP   OCT   NOV   DEC TOTAL PRCNT
    0     0     0     0     0     0     0     0     0     0     0 [illegible]
 
YEAR 1966
  JAN   FEB   MAR   APR   MAY   JUN   JUL   AUG   SEP   OCT   NOV   DEC TOTAL PRCNT
    0     0     0     0     0     0     0     0     0     0     0  128   128 [illegible]
 
YEAR 1967
  JAN   FEB   MAR   APR   MAY   JUN   JUL   AUG   SEP   OCT   NOV   DEC TOTAL PRCNT
    0     0     0     0     0     0     0     0     0     0     0  112   112  18.1
 
YEAR 1968
  JAN   FEB   MAR   APR   MAY   JUN   JUL   AUG   SEP   OCT   NOV   DEC TOTAL PRCNT
    0     0     0     0     0     0     0     0     0     0     0  553   553  21.3
 
YEAR 1969
  JAN   FEB   MAR   APR   MAY   JUN   JUL   AUG   SEP   OCT   NOV   DEC TOTAL PRCNT
    0     0     0     0     0     0     0     0     0     0     0  225   225  09.9
 
TOTAL
    0     0     0     0     0     0     0     0     0     0     0 2280  2280
 
PRCNT
 00.0 00.0 00.0 00.0 00.0 00.0 00.0 00.0 00.0 00.0 00.0 100.0       100.0
 
COUNT BY MOODS SOURCE CODES
   2-  268    3-  230    4- 1344    8-    1   17-  137
 
COUNT BY MOODS INSTRUMENT CODES
   0-  437    1-  268    2-  231   25  1344
 
COUNT BY CLASSIFICATION
  CODE     DIST. STATEMENT   QUALIFICATION   DIST. QUALIFICATION   CLASSIF
"/tmp/2017043970" 50 lines, 1863 characters
```

Figure 17. Page Summary Screen Output Example.

4.1.3.1.12 UTIL: Read Defaults File

The function **createUtil()** registers the callback function **selectReadFile()** to create the "Read Defaults File" pop-up dialog window shown in Figure 18. This dialog window is a standard Motif widget created by the function **XmCreateFileSelectionDialog().** File/directory selection from the scrollable listings, entry of a path/file into the "Selection" textbox and selection of the "Directory" button are handled internally by the widget. The callback function **fileNameAcceptCB()** responds when the "Load" button is selected by calling the function **executeReadDefaultsFile(). executeReadDefaultsFile()** resets MDMS default values to those contained in the newly loaded defaults file via the functions **readMoodsDefaults(), clearMoodsStruct(), clearMoodsSelection()** and **setMoodsSelection().**

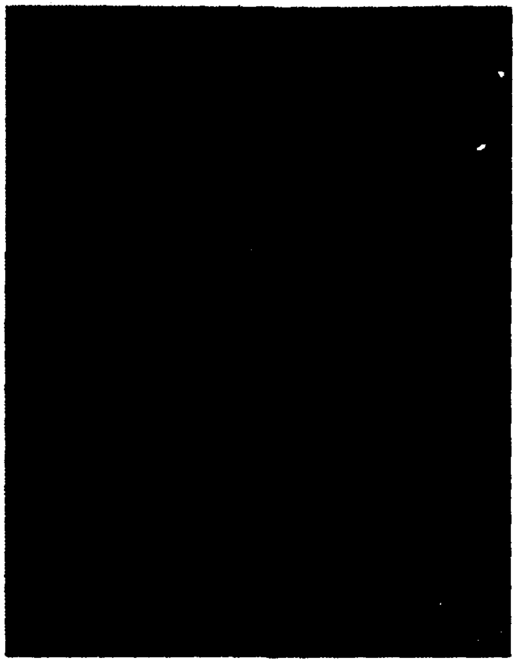


Figure 18. Util Read Defaults Window.

4.1.3.1.13 UTIL: Write Defaults File

The function **createUtil()** registers the callback function **selectWriteFile()** to create the "Write Defaults File" pop-up dialog window shown in Figure 19. **selectWriteFile()**, in turn calls the function **createFileNameDialog()** which registers the additional callback functions **exitDefaultsFile()** and **executeWriteDefaultsFile()** to handle the "Exit" and "OK" buttons, and the callback function **readFileName()** to capture the output filename entered by the user. executeWriteDefaultsFile() collects current values for all default variable members and writes them to a text file.

Figure 19. Util Read Defaults Window.

4.1.3.1.14 HELP

The function **createMenubarOptions()** registers the callback function **moodsOpnHelp()** to create pop-up dialog windows in response to selection of the "Help" menu header. **moodsOpnHelp()**, in turn, registers the callback function **moodsOpnHelpOk()** to process selection of the "OK" button associated with each "Help" pop-up dialog window. **moodsOpnHelp()** registers a recursive callback to itself to process a series of help pop-up dialog windows. Help text is contained in the include file **moodsOpnHelp.h**.

4.1.3.2 The Data Selection Screen

The "Data Selection" portion of the "Main Window" occupies most of the left side of the MDMS Main Window display screen (see Figure 2). The function **dataSelAndDisplay()** is called by the function **CreateTree()** to create both the "Data Selection" and "Selection Status" windows. **dataSelAndDisplay()** relies upon the function **dataSelectDesign()** to construct the "Data Selection" window and its suite of interactive widgets. **dataSelectDesign()** registers the following callback functions to process user selection and entry of key MDMS database query parameters, all of which must be defined prior to any database query:

readCruiseId() - to read the minimum or maximum cruise-ID entered into the "Cruise ID" textbox.
readMonth() - to capture the minimum or maximum month selected from the scrollable "Months" listbox.
readNumParm() - to process the minimum or maximum number of parameters selected via the sliding "Number or Parameters" scale widget.
readLoadDate() - to read the minimum or maximum loading date entered into the "Load Date" textbox.
readSource()* - to capture the minimum or maximum data source code selected from the scrollable "Source" listbox.
readClass()* - to capture the minimum or maximum classification code selected from the scrollable "Classification" listbox.
readInst()* - to capture the minimum or maximum instrument code selected from the scrollable "Instrument" listbox.

*Note: **dataSelectDesign()** employs the function **fillList()** to construct the codes listed in the "Source", "Classification" and "Instrument" listboxes.

The statement...

```
if (moods_struct->query_val->selection_type == MINIMUM)
```

determines if the value retrieved by the callback functions represents a minimum or maximum value for each parameter. **moods_struct->query_val->selection_type** is set by the callback functions **minTogglePress()** and **maxTogglePress()** which are registered by **selectDisplayDesign()**, the function which is ultimately responsible for creating the "Selection Status" window located to the right of the "Data Selection" window. Maximum and minimum values for the key parameters are placed in the structure **moods_struct** where they may be accessed by other MDMS functions.

4.1.3.3 The Selection Status Screen

The "Selection Status" portion of the "Main Window" occupies most of the right side of the MDMS Main Window display screen (see Figure 2). The function **dataSelAndDisplay()** is called by the function **CreateTree()** to create both the "Selection Status" and "Data Selection" windows. **dataSelAndDisplay()** relies upon the function **selectDisplayDesign()** to construct the "Selection Status" window and its suite of interactive widgets. **selectDisplayDesign()** registers the following callback functions to retrieve and display minimum and maximum values for the key parameters defined within the "Data Selection" window:

minTogglePress() - to set data selection mode to minimum in response to clicking the "Minimum" toggle button.
maxTogglePress() - to set data selection mode to maximum in response to clicking the "Maximum" toggle button.
readMinTime() - to retrieve the minimum time from the **moods_struct** structure.
readMaxTime() - to retrieve the maximum time from the **moods_struct** structure.
readMinLon() - to retrieve the minimum longitude from the **moods_struct** structure.
readMaxLon() - to retrieve the maximum longitude from the **moods_struct** structure.
readMinLat() - to retrieve the minimum latitude from the **moods_struct** structure.
readMaxLat() - to retrieve the maximum latitude from the **moods_struct** structure.

The non-callback function **getMonthName()** is invoked to retrieve the minimum or maximum month value from the **moods_struct** structure. The remaining maximum and minimum parameter values are directly retrieved from the **moods_struct** structure and inserted into their text widgets by the function **selectDisplayDesign()**.

4.1.3.4 Remarks

The "Remarks" label button and its accompanying textbox are created and displayed by the function **CreateTree()**. Messages are printed in the "Remarks" textbox by the function **printmesg()** and removed by the function **clearmesg()**.

4.2 CSC-2 - The MDMS Data Management Module (DMM)

Visible functionality of the MDMS is the responsibility of the GUI. The DMM responds to internal requests from the GUI, the DAM or other DMM functions. A request may require data retrieval from the MOODS database. It may require creation of a new process or application of an algorithm on the data after it is retrieved from the database. The DMM may be called upon to read MOODS data from an import file and ingest it into the MOODS database. Effectively, the user is insulated from the functionality of the DMM, seeing only the results which are displayed by the GUI. Any data processing chores performed by the DMM are associated with the need to 1) provide data in specified formats to the GUI and/or DAM, or 2) ingest/retrieve data to/from the supporting MOODS database. The CSU-level design of the DMM is embodied in those functions which interact with the GUI, DAM and the supporting database.

4.2.1 DMM Interaction with the GUI

With two exceptions (see Section 4.2.2, below), all communications with the DMM and its underlying database is invoked by the GUI. At the CSU level, DMM interaction with the GUI is accomplished via the following DMM functions:

InitMoodsStruct() - to allocate memory for the structure **moods_struct** and initialize values. Default data values are obtained from the **moods.defaults** file by the function **readMoodsDefaults()** and passed to GUI functions such as **selectDisplayDesign()** which display the values within its "Selection Status" text widgets. InitMoodsStruct() is also invoked by
exportData2File() - called by the GUI function **importexportOkSelect()** to query retrieve data from database and write it to a file.
readIngest_file() - called by the GUI function **importexportOkSelect()** to import MOODS ADMIN data into database with latitude, longitude and time plus seven additional variables.

executeCompositePlot() - called by **selectParmvsDepth()** to fetch data from the database for use in drawing composite parameter vs. depth plot.
executePrintHeader() - called by outputFormatDialog() to fetch data from the database for use by the Print Header menu option.
executeLogData() - called by **printLogLayout()** to retrieve specific log entries from the database as defined by the MDBA.
executePageSummary() - called by **outputFormatDialog()** to retrieve header information from the database for use by the Page Summary menu option.
getObsData() - called by **distributionDraw()**, **exposeDistribution()**, **exposeHistogram()** and **exposeAccl()** to read observation data from database for distribution, histogram and accumulation plots.
lltMapFuncs() - called by **distributionDraw()** and **exposeDistribution()** to read geographic coastline data from database for plotting on a registered grid.

4.2.2 DMM Interaction with the DAM

As indicated in Figure 3, the DMM and DAM are not routinely required to communicate directly in order to support MDBA-specific actions relative to database maintenance. In most cases, the communication is accomplished indirectly; i.e., the GUI serves as the messenger between the DAM and the DMM. The following DMM functions are exceptions in that they do communicate directly with the DAM:

moodsUpdateQltvInfo() - called by the DAM function **changeTable()** to update the MOODS database qualitative information table and make the appropriate entries in the MOODS Log table.
moods_brws_2() - called by the DAM function **chk_clas_acc()** to compare access privileges of user with selected classification code range.

4.2.3 DMM Interaction with the MOODS Database

DMM functions that interact with the MOODS database and their purpose are as follows:

commonHeaderRead() - to read header information from the database.
llt7_rd() - to read llt records from the database.
lltn_opn() - to open the MOODS database.
moodsUpdateQltvInfo() - to update the MOODS database qualitative information table and makes the appropriate entry in the MOODS Log.
llt7_wr() - to write a primary MOODS profile data record to the database.
lltn_wr_as() - to write an associative record to the MOODS database for an ingested dataset.

4.3 CSC-3 - The MDMS Data Administration Module (DAM)

The purpose of the DAM is to support 1) MOODS Database Administrator (MDBA) control of the MDMS and 2) to exclude user access to unauthorized data. User access criteria can only be established by the MDBA, which emphasizes the critical security responsibilities of the MDBA. The DAM interacts directly with the GUI in support of database and software maintenance requirements. Aside from the two exception noted in Section 4.2.2, there is no direct link between the DAM and the DMM. Instead, interaction between the DAM and the DMM is accomplished via the GUI.

4.3.1 DAM Interaction with the GUI

At the CSU level, the following DAM functions interact with the GUI to perform the tasks indicated:

getUlogin() - called by **main()** to obtain user identification of the current process.
getDba() - called by **main()** to identify the database creator.
checkAccess() - called by **importexportOkSelect()** and **exportData2File()** to ensure verify user access permissions for file read/write operations.
changeTable() - called by **insCodeType()**, **updateCodeType()** and **delCodeType()** to make necessary changes to MDBA-controlled parameter values (classification, instrument and source).

4.3.2 DAM Interaction with the DMM

(see Section 4.2.2)

5 MDMS DATA

5.1 The MOODS Database

The MOODS database is described in Appendix C.

5.2 MDMS User Configuration Data

User configuration data is contained in the **moods.defaults** file. An example **moods.defaults** file can be found in Appendix G.

6 MDMS DATA FILES

The MDMS retrieves data from the MOODS database, data import files and the user configuration file. There are no other data requirements for the MDMS.

7 REQUIREMENTS TRACEABILITY

MDMS functional requirements (MFR) and design requirements (MDR) have been defined for the MDMS. MFR/MDR requirements are also indicated in paragraphs 3.2.1, 3.2.2 and 3.2.3 for cross-reference and traceability. CSC responsibility for achieving each MFR and MDR are indicated parenthetically in the following descriptions for each MFR and MDR.

MDMS FUNCTIONAL REQUIREMENTS (MFR)

MFR1: Operate in an interactive manner; i.e., displays must be interactive. (GUI)
MFR2: Interactively identify geographical region boundaries. (GUI)
MFR3: Interactively identify time and date ranges. (GUI)
MFR4: Interactively identify specific classification codes and ranges of same. (GUI)
MFR5: Interactively identify specific cruise identification numbers and ranges of same. (GUI)
MFR6: Interactively identify specific data source codes and ranges of same. (GUI)
MFR7: Interactively select specific month constraints and ranges of same. (GUI)
MFR8: Interactively enter specific data loading dates and ranges of same. (GUI)

MFR9: Interactively toggle between selection of minimum and maximum retrieval parameter criteria. (GUI)
MFR10: Interactively select combinations of temperature, salinity and/or sound speed for processing. (GUI)
MFR11: Provide on-line help to users. (GUI)
MFR12: Provide on-line addition, deletion and update capability to the MDBA for classification codes, cruise identification numbers and data source codes. (GUI/DAM)
MFR13:On-line data import capability for MDBA. (GUI/DMM/DAM)
MFR14: On-line data export capability for users. (GUI/DMM)
MFR15: Error feedback via message facility. (GUI/DMM/DAM)
MFR16: Capability for multiple configuration/initialization files. (CSCI)
MFR17: Interactively indicate completion of retrieval criteria development and commencement of data retrieval operation based on the indicated criteria. (GUI/DMM)
MFR18: Overlay data distribution plots as points on a geographic map window. (GUI/DMM)
MFR19: Produce histogram display of data density. (GUI/DMM)
MFR20: Display depth versus parameter (temperature, salinity, sound speed) plots. (GUI/DMM)
MFR21: Display temperature versus salinity plots. (GUI/DMM)
MFR22: Output header information to printer, screen or file. (GUI/DMM)
MFR23: Maintain log of user activity with ability to print same to printer, screen or file.(GUI/DMM)
MFR24: Output statistical summary of data retrieved from database with ability to print same to printer, screen or file. (GUI/DMM)
MFR25: Provide MOODS data editing capability. (independent of MDMS)
MFR26: System must be able to graphically plot temperature and salinity profiles onto a geographic grid or X-Y plot. (GUI)
MFR27: Access to MOODS ASCII format. (DMM)
MFR28: Access to MOODS ADMIN format. (DMM)
MFR29: Access to Coastlines/Shorelines. (DMM)
MFR30: Selectively evaluate and/or edit MOODS data. (GUI/DMM)
MFR31: Selectively retain subsets of environmental data after evaluation and/or editing procedures have been performed. (DMM)
MFR32: Interactive termination of application. (GUI)
MFR33: Interactive reset of data retrieval parameters. (GUI)

MDMS DESIGN REQUIREMENTS (MDR)

MDR1: The MDMS must operate as a stand-alone system. (CSCI)
MDR2: The MDMS must operate within the UNIX operating system environment. (CSCI)
MDR3: The MDMS must execute within the X-Windows client-server model. (CSCI)
MDR4: Window displays must incorporate the Open Software Foundation (OSF) Motif Widget Library. (CSCI)
MDR5: An independent relational database management system (rdbms) specifically for MOODS utilization. (DMM)
MDR6: The Naval Environmental Operational Nowcast System (NEONS) will be incorporated as the interface to the rdbms. (DMM)
MDR7: The MDMS must contain internal links to the MOODS rdbms for data import/export/ retrieval. (DMM)
MDR8: The MDMS must incorporate detection of user privilege levels based on data classification codes. (GUI)

8 NOTES

A glossary of terms is contained in Appendix A. A listing of acronyms is contained in Appendix B. Appendix C is the MDMS RDBMS Specification. Appendix D lists MOODS RDBMS functional (DBFR) and design (DBDR) requirements. Appendix E discusses the MDMS development environment. Appendix F is a listing of MOODS structures. Appendix G provides an explanation of the user default file.

APPENDIX A. GLOSSARY OF TERMS

Computer Software Configuration Item (CSCI) - a software application or a major component thereof.

Computer Software Component (CSC) - a top level functional module within a computer software configuration item (CSCI). CSC's are generally considered to be one structural level below the CSCI.

Computer Software Unit (CSU) - low level software modules, usually at the function or subroutine level that perform specific functions within a CSC.

Data Administration Module (DAM) - the MDMS module responsible for MDBA security and administrative maintenance functions.

Data Management Module (DMM) - the MDMS module responsible for database access.

Graphical User Interface (GUI) - the MDMS module responsible for interfacing with the user and controlling the graphical and display functionality of the MDMS.

Widget - "... a graphic device capable of receiving input from the keyboard and the mouse and communicating with an application or another widget by means of a callback. Every widget is a member of only one class and always has a window associated with it." (from: OSF/Motif Programmer's Guide, Rev. 1.1, Open Software Foundation, Prentice Hall, Englewood Cliffs, NJ, 1991, p. GL-13.)

APPENDIX B. LIST OF ACRONYMS

ANSI - American National Standards Institute
CAST - Center for Air Sea Technology
CSC - Computer Software Component
CSCI - Computer Software Configuration Item
CSU - Computer Software Unit
DAM - Data Administration Module
DBDR - Database Design Requirement
DBFR - Database Functional Requirement
DMM - Data Management Module
DOD - Department of Defense
GUI - Graphical User Interface
IDEAS - Interactive Data Editing and Analysis System
MDBA - MOODS Database Administrator
MDMS -MOODS Data Management System
MDR - MDMS Design Requirement
MFR - MDMS Functional Requirement
MOODS - Master Oceanographic Observation Data Set
MSU - Mississippi State University
NASA - National Aeronautics and Space Administration
NAVOCEANO - Naval Oceanographic Office
NEONS - Navy Environmental Operational Nowcast System
OSF - Open Software Foundation
RDBMS - Relational Database Management System
SQL - Structured Query Language

APPENDIX C. THE MOODS RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS) SPECIFICATION

The MDMS accesses the MOODS RDBMS, a modified NEONS database created specifically to support the CSCI. The MOODS database exists as a dedicated external and independent element of the system, accessible via embedded SQL queries to the underlying RDBMS engine. The MDMS queries the database and retrieves requested data from it by calling NEONS software library functions. NEONS provides a data model for all generic data types (grid, volume, latitude/longitude/time (LLT), line, image and geographical); however, the MDMS only uses a modified version of the latitude/longitude/time data type to store MOODS profiles, plus the geographical data type which stores coastlines and ocean bathymetry datasets. MOODS profile datasets can be retrieved by the following parameters:

> latitude, longitude, date/time, month, water depth, parameter, source code, instrument, classification code, and cruise identification number.

These parameters represent an enhancement of NEONS capability to meet MDMS requirements. The NEONS latitude/longitude/time data type allows queries only by latitude, longitude and date/time. The additional parameters (month, water depth, parameter, source code, instrument, classification code, and cruise identification number) were moved from the RDBMS binary large object storage format to allow faster dataset specification and retrieval. The MDMS-specific version of the LLT data type is referred to as "LLTN_7" because of the seven additional query parameters. The MDMS version of NEONS also incorporates special relational tables to support security classification codes and a MOODS transaction log.

Further information about NEONS, its structure and use is available in the NEONS design document ["Database Design Document for the Naval Environmental Operational Nowcast System", Version 3.5, Naval Research Laboratory (NRL), Monterey, CA 93943-5006]. In addition, the NEONS installation package includes source code and manual pages for each major functional element. NEONS version control is managed by the Naval Research Laboratory, Monterey, CA.

APPENDIX D. FUNCTIONAL AND DESIGN REQUIREMENTS FOR THE MOODS RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)

DATABASE FUNCTIONAL REQUIREMENTS (DBFR):

DBFR1: Database must be capable of data retrieval.
DBFR2: Data ingestion into the rdbms is to be accomplished either internally or externally to the MDMS.
DBFR3: The searchable database attributes must include month, water depth, parameter, source code, instrument type, classification and cruise identification number.

DATABASE DESIGN REQUIREMENTS (DBDR):

DBDR1: RDBMS engine is Empress.
DBDR2: RDBMS model is the Naval Environmental Operational Nowcast System (NEONS) Version 3.5.1.
DBDR3: Database must be compatible with NAVOCEANO Program Modernization Initiative (PMI).

APPENDIX E. THE MDMS DEVELOPMENT ENVIRONMENT

The MDMS has been developed in a Sun Microsystems SparcStation model 10 computer hardware environment. The operating system was SUNOS version 4.1.3, including the resident SUN C compiler which was used to write the MDMS software code. Some minor elements of NEONS have been written in FORTRAN77 (Sun FORTRAN77 version 1.4). Graphics support is provided by UNIRAS ag/X Toolmaster version 6v3b. The RDBMS engine is Empress version 6.2. The windowing environment consists of X-Windows version X11 R5 and the OSF Motif widget set version 1.3.

APPENDIX F. MOODS STRUCTURES

A C structure is a collection of data variables, pointers or other structures grouped together for the convenience of using a single variable name to reference or identify the whole group. The use and behavior of structures is covered in any credible C programming language tutorial or textbook.

1. **The NAVO Structure** - comprises the critical data framework of the MDMS application. The **NAVO** structure and its internal structures are defined as follows:

```
typedef struct {
        STATUS_BOARD_WIDGETS *status_board_widgets;
        QUERY_VAL      *query_val;
        TIME_LOC         *LocTime;
        Widget           region, globe_map;
        char             seq_type[31];
        char             vrsn_name[31];
        char             defaultsFile[120];
        } NAVO;
```

Structures Internal to the NAVO Structure

```
typedef struct {
        Widget lat_min_text, lat_max_text;
        Widget lon_min_text, lon_max_text;
        Widget time_min_text, time_max_text;
        Widget class_min_text, class_max_text;
        Widget month_min_text, month_max_text;
        Widget parm_min_text, parm_max_text;
        Widget cruise_min_text, cruise_max_text;
        Widget inst_min_text, inst_max_text;
        Widget source_min_text, source_max_text;
        Widget load_min_text, load_max_text;
        Widget min_toggle, max_toggle;
        } STATUS_BOARD_WIDGETS;

typedef struct {
        double lat_min_val, lat_max_val;
        double lon_min_val, lon_max_val;
        DATE  time_min_val, time_max_val;
        int   hour_min_val, hour_max_val;
        int   mnt_min_val, mnt_max_val;
        int   sec_min_val, sec_max_val;
        long  class_min_val, class_max_val;
        long  month_min_val, month_max_val;
        long  parm_min_val, parm_max_val;
        long  cruise_min_val, cruise_max_val;
        long  inst_min_val, inst_max_val;
        long  source_min_val, source_max_val;
        long  load_min_val, load_max_val;
        Boolean selection_type;
        } QUERY_VAL;
```

```
typedef struct {
        DATE         start_date;
        DATE         end_date;
        double       min_lat;
        double       min_lon;
        double       max_lat;
        double       max_lon;
        int          start_hour;
        int          end_hour;
        int          start_min;
        int          end_min;
        int          start_sec;
        int          end_sec;
        } TIME_LOC;
```

Note: The Widget structure is proprietary to the XLib/MOTIF libraries.

2. **The NAVO_DATE Structure** - defines the elements which comprise date and time within the MDMS.

```
typedef struct {
        int hour;
        int min;
        int day;
        int month;
        int year;
        } NAVO_DATE;
```

3. **The DISTRIBUTION Structure** - groups all variables required to plot a histogram or accumulation chart.

```
typedef struct {
        float        lat_deg;        /* latitude  interval */
        float        lon_deg;        /* longitude  interval */
        int          lon_binsize;    /* longitude cell count */
        int          lat_binsize;    /* latitude cell count */
        Boolean      grid;           /* To indicate if grid is needed */
        char         coast_type[15]; /* 3km, 8km, 20km */
        char       title1[50];   /* Plot type title */
        char       title2[50];   /* User-supplied title */
        int          dev_type;       /* Screen/window */
        int          plot_type;      /* Plot/Histogram/Accumulation */
        int  .       rec_num;        /* Num of Obs Read */
        Boolean      thereisaacclpix;/* Indicates presence of Pixmap */
        Boolean      thereisahistpix;/* Indicates presence of Pixmap */
        Boolean      thereisapix;    /* Indicates presence of Pixmap */
        Pixmap               pixmap;              /* Pixmap to store graphics */
        XWindowAttributes pix_at;    /* Store dimensions of Pixmap */
        XWindowAttributes hist_pix_at; /* Store dimensions of Pixmap */
        XWindowAttributes accl_pix_at; /* Store dimensions of Pixmap */
        Pixmap              hist_pixmap;  /* Pixmap to Histogram */
        Pixmap              accl_pixmap;  /* Pixmap to Accumulation plot */
        float          pix_width;      /* Size of Display Pixel */
        float          *hist_array;    /* To keep count in cells */
        Boolean        dataset_selected;/* Avoid reselecting data for Hist*/
        Widget              coast_text;   /* Indicate choice Made */
        NAVO                *moods_struct; /* Global Structure */
        Window              accl_window;   /* Window ID */
        Window         hist_window;    /* Window ID */
        Window         plot_window;    /* window ID */
        } DISTRIBUTION;
```

4. **The PARMVSDEPTH Structure** - groups all variables required to plot a parameter vs. depth chart.

```
    typedef struct {
        float          min_x;          /* Min x-axis value */
        float          max_x;          /* Max x-axis value */
        float          min_y;          /* Min y-axis value */
        float          max_y;          /* Max y-axis value */
        char       title[50];     /* graph title */
        float          **depth;        /* depth Values            */
        float          **data_array;  /* Data Values     */
        int            *prof_points;  /* Points Per Profile */
        int            prev_max_count;        /* Maximum profiles */
        int            max_count;     /* Maximum profiles */
        int            parm_type;     /* Parameter type  */
        int            total_obs;      /* Total Recs        */
        NAVO                *moods_struct; /* Global Structure */
        Boolean        thereisacomppix;/* Boolean Flag           */
        Boolean        dataset_selected;/* Boolean Flag    */
        Pixmap         comp_pixmap; /* Pixmap            */
        Window              window;                 /* Window ID        */
        int            dev_type;       /* Screen/window */
        } PARMVSDEPTH;
```

5. **The LOGSTRUCT Structure** - groups the necessary variables to output a portion of the MDMS transaction log.

```
    typedef struct {
        int   log_type;
        Widget list;
        Widget to_screen;
        Widget to_file;
        Widget to_printer;
        Widget name_text;
        Widget min_date_text;
        Widget max_date_text;
        Widget msg_text;
        char   userName[20];
        DATE   min_date;
        DATE   max_date;
        long   func_id;
        long   min_func_id;
        long   max_func_id;
        char   filename[80];
        } LOGSTRUCT;
```

6. **The SMRYPAGE Structure** - groups the necessary variables to output a summary page.

```
    typedef struct {
        float min_lat;
        float min_lon;
        float max_lat;
        float max_lon;
        struct {
            int yr;
            int ttl_yr_cnt;
            struct {
                int mo;
                long ttl_mo_cnt;
                } mo_bin[12];
            }  yr_bin[MAX_CNT];
        struct {
            long src_code;
            long ttl_src_cnt;
            }  src_bin[MAX_CNT];
        struct {
            long inst_code;
            long ttl_inst_cnt;
            }  inst_bin[MAX_CNT];
        struct {
            int  clas_code;
            long ttl_clas_cnt;
            long ttl_qual_cnt;
            long ttl_dist_cnt;
            long ttl_dist_qual_cnt;
            }  clas_bin[MAX_CLAS_CNT];
        }  SMRY_PAGE;
```

7. **The IMPORTEXPORT Structure** - contains the necessary variablers to import or export a data file.

```
    typedef struct {
        NAVO        *moods_struct;  /* Global Structure */
        char   filename[80];    /* Filename to store header */
        int      type;          /* IMPORT,EXPORT, or EXPORT_IDEAS  */
        Widget ok;              /* Ok Widget */
        } IMPORTEXPORT;
```

APPENDIX G. THE MDMS DEFAULT CONFIGURATION FILE

Default values for parameters appearing within the "Selection Status" textboxes of the MDMS main display may be maintained in a user-created default file. The user may specify the file containing default values by including the "-f" flag, followed by the filename, when launching the MDMS (example: moods -f myfile). If the default file is not indicated on the command line, the MDMS will look for the file **moods.defaults** in the current directory. If the **moods.defaults** file cannot be located, parameter boxes within the "Selection Status" portion of the main display will be blank at startup. The format for a default file is as follows:

```
Latitude       21.3        30.0
Longitude       -121.0       -109.0
Date/Hour       19621207:6:0:0 19931008:12:0:0
Classification  1100010      1100010
Month          3           3
Parms          2           3
Cruise-Id       300         300
Instrument      25          25
source         4           4
load-date       19920616     19920616
```

The values contained in a default file are ordered in a top-to-bottom sequence according to their appearance within the "Selection Status" portion of the main display. The format is free form except that labels and numerical values must be separated by at least one space and each labeled line must end in a carriage return. For each labeled category, there are two numerical entries, a minimum and maximum value, on each line. The first numerical entry is the minimum value. The second numerical value is the maximum value.

DISTRIBUTION LIST

1. Commanding Oficer, Code N3211
   Naval Oceanographic Office
   Stennis Space Center, MS 39529
   (10 copies)

2. Technical Director
   Code OOT
   COMNAVMETOCCOM
   Stennis Space Center, MS 39529

3. Occeanographer of the Navy
   U.S. Naval Observatory
   34th and Massachussetts Avenue
   Washington, DC 20392

4. Space and Naval Warefare
   Systems Command
   Code PMW175-3B
   2451 Crystal Drive
   Arlington, VA 22245-5200

5. Defense Technical Information Center
   Building 5, Cameron Station
   Alexandria, VA 22304-6145
   (2 copies)

6. Director, Sponsored Programs Administration
   Mississippi State University
   P.O. Box 6156
   Mississippi State, MS 39762

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. Agency Use Only (Leave blank).	2. Report Date. AUGUST 1994	3. Report Type and Dates Covered. TECHNICAL NOTE

4. Title and Subtitle.
DESIGN DOCUMENT FOR THE MOODS DATA MANAGEMENT SYSTEM (MDMS) VERSION 1.0

5. Funding Numbers.
Program Element No.
Project No.
Task No.
Accession No.

6. Author(s).
RAMESH KRISHNAMAGARU M.S. FOSTER
CHERYL CESARIO VISHNUMOHAN DAS

7. Performing Organization Name(s) and Address(es).
MISSISSIPPI STATE UNIVERSITY
CENTER FOR AIR SEA TECHNOLOGY
BUILDING 1103, ROOM 233
STENNIS SPACE CENTER, MS 39529

8. Performing Organization Report Number.
CAST TECHNICAL NOTE 6-94

9. Sponsoring/Monitoring Agency Name(s) and Address(es).
NAVAL OCEANOGRAPHIC OFFICE (CODE N3211)
STENNIS SPACE CENTER, MS 39529

10. Sponsoring/Monitoring Agency Report Number.
CAST TECHNICAL NOTE 6-94

11. Supplementary Notes.
Research performed via Mississippi Research Consortium under NASA Procurement Office Contract NAS13-330, Delivery Order #53.

12a. Distribution/Availability Statement.
Approved for public release; distribution is unlimited

12b. Distribution Code.

13. Abstract *(Maximum 200 words).*

The MOODS Data Management System (MDMS) provides access to the Master Oceanographic Observation Data Set (MOODS) maintained by NAVOCEANO. This manual provides the design document and database specification for the MDMS.

14. Subject Terms.
(U) DESIGN DOCUMENT (U) MOODS (U) MDMS
(U) CAST (U) GUI (U) NAVOCEANO (U) NEONS

15. Number of Pages. 47

16. Price Code.

17. Security Classification of Report.	18. Security Classification of This Page.	19. Security Classification of Abstract.	20. Limitation of Abstract.
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	